Descrição do Time de Futebol de Robôs da Equipe SIRSoccer para a Categoria *IEEE Very Small Size Soccer*

Johnathan Fercher da Rosa[2], Lucas Borsatto Simão[1], Hebert Luiz Cabral da Silva[1],
Felipe de Mello Amaral[1], Pedro Mello[1],
Alberto Torres Angonese[1,2] e Eduardo Krempser da Silva[1,3].

***Resumo*— Neste artigo são apresentados os resultados do trabalho realizado em 2016 pela equipe SIRSoccer de futebol de robôs, categoria *Very Small Size Soccer - VSSS*, para participação na XIII Competição Brasileira de Robótica - CBR e na XIV Competição Latino Americana de Robótica - LARC. Aqui estão descritos os trabalhos de todas as etapas de construção do time de futebol. Uma descrição mais detalhada é apresentada nas seções: *Framework*, Sistema e Controle, Transmissão e *Hardware*, uma vez que essas foram as partes com maior evolução desde a primeira participação da equipe em 2013.**

I. INTRODUÇÃO

A Faculdade de Educação Tecnológica do Estado do Rio de Janeiro - FAETERJ/Petrópolis, possui o curso superior de Tecnologia da Informação e de Comunicação, em que além das disciplinas tecnológicas previstas no currículo, os alunos também participam de outras atividades, de forma a contribuir para a formação acadêmica. O Laboratório de Sistemas Inteligentes e Robótica - SIRLab, surge com a ideia de proporcionar um ambiente para prática e estudo em computação, robótica e inteligência computacional.

A IEEE *Very Small Size Soccer - VSSS* é uma categoria de futebol de robôs [1] que proporciona o estudo em diversas áreas. O time de robôs que jogam futebol trata-se de um sistema autônomo com um servidor e três agentes que executam ações determinadas por este servidor. As informações de jogo chegam ao servidor por meio de imagens, estas fornecidas por uma câmera que fica instalada a $2m$ ou mais da base do campo. As informações são processadas pelo servidor e em seguida um comando é enviado aos robôs por rádio frequência.

O sistema de um time de futebol de robôs IEEE VSSS é cooperativo e atua em um ambiente altamente dinâmico, o que faz do seu desenvolvimento algo muito complexo. As seções deste artigo estão divididas conforme as linhas de estudo necessárias para o desenvolvimento do trabalho. No desenvolvimento do projeto foram necessários estudos em visão computacional, sistemas e controle, redes e eletrônica. Assim, o trabalho feito será apresentado partindo do novo *Framework* criado pela equipe até o *hardware* do robô.

*Este trabalho é apoiado pelo Laboratório de Sistemas Inteligentes e Robótica - SIRLab e pela Faculdade de Educação Tecnológica do Estado do Rio de Janeiro - FAETERJ

[1]Faculdade de Educação Tecnológia do Estado do Rio de Janeiro - FAETERJ
[2]Instituto Militar de Engenharia - IME
[3]Laboratório Nacional de Computação Científica - LNCC

II. FRAMEWORK

Atualmente o laboratório vem trabalhando na integração das ferramentas desenvolvidas durante esses anos de trabalho, foi desenvolvido, inspirado na arquitetura fornecida para os competidores das categorias Small Size League (SSL) [2] e RoboCup 2D Soccer Simulation, programas separados e fechados de forma a os novos integrantes do laboratório terem que se preocupar apenas com a construção e aprimoramento do software de inteligência.

Utilizando as bibliotecas Protobuf e ZMQ, foi criado um protocolo de comunicação, afim de padronizar os fluxos de entrada e saída de dados entre os seguintes programas: Software de Visão Computacional (VSS-Vision), Simulador da categoria IEEE VSS (VSS-Simulator), (Visualizador de Estados) VSS-Viewer e (Núcleo de Estratégia) VSS-SampleStrategy. Atualmente uma estratégia/inteligência escrita aos moldes do VSS-SampleStrategy funciona sem necessidades de adaptação com os demais projetos, assim facilitando a integração e o desenvolvimento.

A. VSS-Vision

O novo sistema de visão computacional VSS-Vision utiliza a biblioteca OpenCV [4] e faz uso do padrão de cores HSV para captação e filtragem de imagens. A OpenCV é uma biblioteca livre, sob a licença de código aberto BSD (*Berkeley Software Distribution*). A biblioteca pode ser implementada em Python, C++ e outras linguagens, porém, dado que todo o restante do sistema está implementado em C++, optou-se por essa linguagem [5].

O programa localiza na imagem as cores desejadas (Figura **??**), e faz a distinção dos objetos do jogo. E para o tratamento das imagens captadas são usados alguns filtros da biblioteca [6]. O sistema obteve avanços, uma vez que com o uso do HSV ao invés do RGB, os problemas relacionados ao gradiente de luminosidade foram resolvidos. Pode ser visto em [7], um estudo comparativo de diversos sistemas de cores aplicados à segmentação por cor, onde o modelo HSV apresentou informações de cor e luminância mais descorrelacionadas que no modelo RGB. Com isso, o modelo HSV mostra-se mais adequado para segmentação baseada em cores. E após a mudança no programa, o sistema está mais robusto a variações luminosas.

Fig. 1. VSS-Vision em Execução

B. VSS-Simulator

O simulador criado pela equipe do laboratório SIRLab para a categoria VSSS (Figura 2) tem o objetivo de facilitar a criação de estratégias que podem ser implementadas pelos robôs. O padrão de projeto utilizado garante que, mesmo que o código seja feito pelo simulador, ele irá funcionar também para a plataforma física da categoria, de forma que é possível que os integrantes realizem seus progressos na estratégia e apenas tenham que migrar o código para que possa ser executado nos robôs.

Fig. 2. Janela Gráfica de Execução do VSS-Simulator

Isso é possível devido a modulação do projeto, que é dividido em partes físicas, gráficas e de estratégia. A parte física fornece os mesmos dados que a plataforma com a câmera, de modo que, garantido que os dados vão abastecer as outras partes da mesma maneira, é possível o desenvolvimento independente.

O simulador é desenvolvido com o apoio das bibliotecas de OpenGL [8], responsáveis pela parte gráfica, e com a versão 2.83 da Bullet Physics [9], que tem a função de simular toda a física da plataforma. A Bullet funciona com base nos intervalos de tempo que um ciclo do OpenGL leva para poder ser processado, fazendo com que o processamento físico seja sincronizado com o processamento gráfico, sendo realizado logo depois. Os robôs são criados via Bullet, que então fica responsável pela sua movimentação. Essa parte é executada com base nos dados que a estratégia do simulador define, determinando a velocidade que as rodas do robô devem atingir para que possa alcançar o ponto de destino desejado, de forma que fica a critério do desenvolvedor determinar qual a rota que o robô irá adotar.

O desenvolvimento da plataforma de simulação requere alguns cuidados, pois a própria Bullet possui um sistema de veículos próprios, mas que teve que ser modificado pela equipe para que pudesse ser compatível ao modelo da categoria VSSS. Um exemplo técnico disso é o sistema de movimentação dos veículos dentro da biblioteca, que ocorre com base em um sistema de coordenadas relativa ao sistema cartesiano externo, e não relativo ao próprio veículo. Como a velocidade do veículo é sempre incrementada de um valor baseado em um determinado impulso (proveniente dos intervalos do ciclo), isso causa alguns problemas, como quando se acrescenta velocidade ao veículo, mas o vetor unitário que determina sua parte frontal tem as suas duas componentes X e Y negativas, fazendo com que, ao invés de a velocidade seja incrementada, ela acabe diminuindo.

Assim, testes vêm sendo feitos na parte da estratégia. Comparações são feitas entre os resultados obtidos no simulador e no sistema físico e, logo em seguida, são feitas modificações no sistema físico, ou um novo método é testado no simulador.

III. SISTEMA E CONTROLE

Os robôs da categoria IEEE *Very Small Size Soccer* (VSSS) são do tipo diferenciais [10] e não holonômicos (Figura 3). Sendo a holonomia o conjunto de vínculos relacionados a mecânica onde há dependências apenas relacionadas às coordenadas espaciais que definem o sistema. E o termo diferencial faz referência ao modelo de robôs com duas rodas que não são capazes de se movimentar em todas as coordenadas espaciais sem que mudem suas angulações.

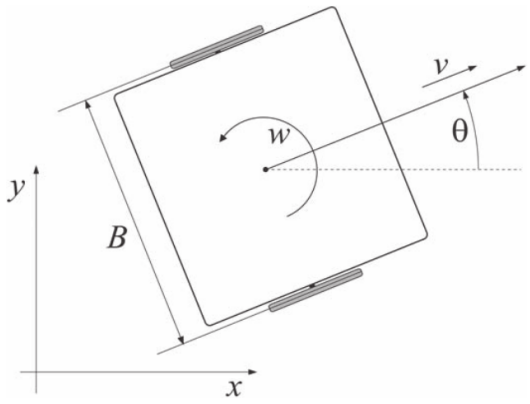

Fig. 3. Modelo *Differential Driver*

As equações cinemáticas de um robô da categoria VSSS são equivalentes ao modelo de um uni ciclo. Robôs com tal arquitetura tem uma descrição não holonômica e cinemática de forma [11]:

$$\dot{q} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} cos\theta & 0 \\ sen\theta & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} v \\ \omega \end{pmatrix} \tag{1}$$

Sendo $\dot{q}$ a pose atual do sistema, $\dot{x}$ e $\dot{y}$ as componentes da velocidade tangencial em $\dot{x}$ e $\dot{y}$, $\dot{\theta}$ a angulação do robô e v e ω as velocidades tangencial e angular, respectivamente. Através da velocidade tangencial calculada no modelo (1), é feita a estimação das velocidades tangenciais para cada uma das rodas pelas equações (2,3).

$$v_e = v - \frac{\omega B}{2} \tag{2}$$

$$v_d = v + \frac{\omega B}{2} \tag{3}$$

Para concluir-se que o sistema se comporta como deveria é necessário que este alcance a chamada referência. Sendo assim, enquanto uma ou mais variáveis de saída não conseguem alcançar a referência, ao longo do tempo, um controlador manipula as entradas do sistema para obter o efeito desejado nas saídas. Esse controlador é aqui representado por $u(t)$, e o erro a ser corrigido para chegar-se convergência do sistema por $e(t)$. O controle PID é dado então pela equação (4).

$$u(t) = k_p.e(t) + k_d.\frac{de(t)}{dt} + k_i.\int_0^t e(t)dt \tag{4}$$

Sendo k_p, k_d e k_i parâmetros de ajuste do controle.

IV. TRANSMISSÃO

A rede do time de futebol de robôs da SIRSoccer possui uma topologia do tipo estrela. O coordenador é configurado em modo broadcast, e os pacotes são transmitidos em massa para os três robôs, que fazem a interpretação do pacote e executam apenas o trecho da mensagem destinado a cada um deles.

A rede é composta por três receptores configurados como *end-point*, e um transmissor principal configurado como coordenador [12]. A frequência operante dos rádios XBee S3 (Figura 4) podem ser definidas de $902Mhz$ (IEEE 802.15.4) até $928Mhz$ (IEEE 802.15.4), diferente da versão anterior usada pela equipe em suas primeiras participações, que operava em $2.4Ghz$ (IEEE $802.11x$) como frequência padrão.

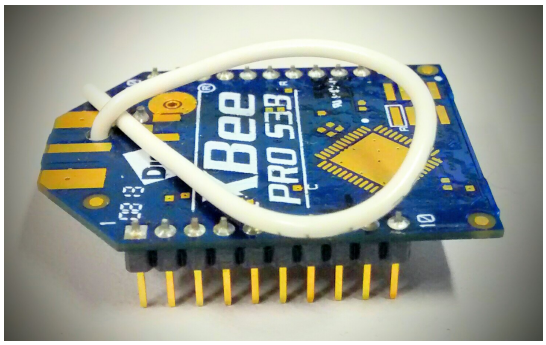

Fig. 4. XBee Pro S3B

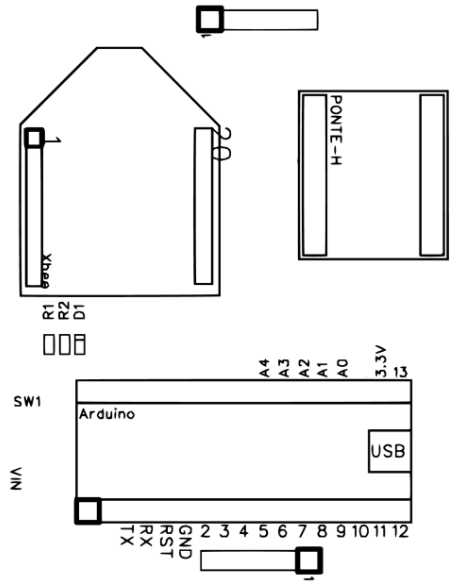

Fig. 5. Componentes do Hardware dos Robôs SIRSoccer

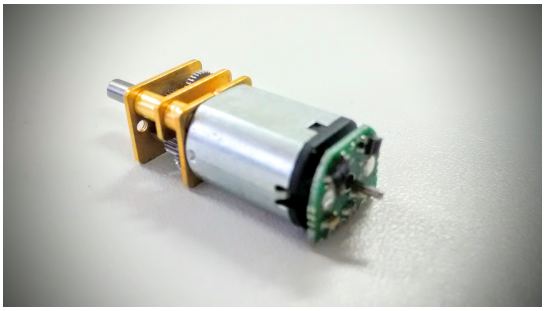

Fig. 6. Motor de relação 1:50 com encoder

V. HARDWARE

Os componentes base para o desenvolvimento da plataforma foram o rádio Digi Xbee S3B [13], o microcontrolador Arduino Micro [14], o circuito integrado da Toshiba TB6612FNG [15], os micro-motores da pololu [16] e seus *encoders* [17] ópticos para micro motores. Foi então projetada uma placa dedicada a interconexão dos componentes básicos da plataforma (Figura 5), de modo a poupar espaço e a proporcionar melhor aproveitamento de baterias evitando que componentes que não estão sendo usados promovam gastos excessivos no sistema eletrônico.

Para receber as informações do servidor, o sistema robótico dispões de um rádio RF XBee S3B, circuito este responsável por transpor a comunicação de baixo nível lógico do XBee para o nível lógico do micro-controlador. A decodificação da informação recebida é feita pelo microcontrolador Arduino Micro, que vai enviar os comandos de movimentação para uma ponte-h [18] dupla da Toshiba, modelo TB6612FNG. Por fim, esta será responsável por transformar os comandos básicos de movimentação em pulsos para os motores [19].

Assim, com o espaço poupado pela utilização de uma placa única, foi possível usar duas baterias que somam um total de 2000mAh e operam em uma voltagem de 7,4V. E ainda foi possível utilizar *encoders* (Figura 6) nos micro motores para que através das leituras de velocidade de rotação fosse possível implementar um controle embarcado.

A motivação para criação desta placa advém de experiencias da equipe em competições anteriores. O uso de *shields* [20] pré-moldados com pinagens de fácil integração entre *drivers* de motor, dispositivos de rádio frequência e microcontroladores foram essenciais para que a equipe pudesse em um primeiro momento ter outras preocupações. No entanto, com o desenvolver do projeto, a busca por uma melhor desempenho dos robôs (Figura 7 tem se mostrado essencial.

VI. CONCLUSÃO

Durante os trabalhos em 2016, a equipe SIRSoccer estudou técnicas e modelos para aperfeiçoar seu time de futebol de robôs *Very Small Size Soccer*. Dentre os avanços obtidos este ano, destaca-se o início de estudos mais aprofundados em sistemas e controle, a criação de uma placa e termino

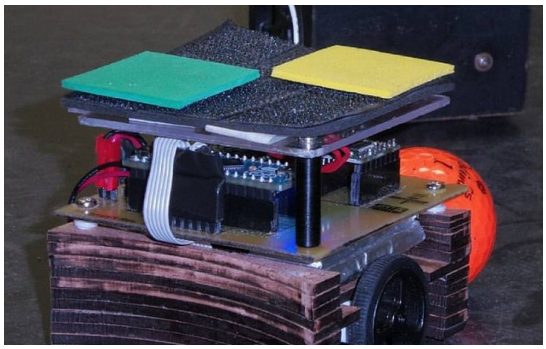

Fig. 7. Robô SIRSoccer

do desenvolvimento do simulador. Planos já estão sendo traçados para as futuras participações da equipe na LARC, e este ano espera-se obter resultados ainda mais satisfatórios que nas edições anteriores.

VII. AGRADECIMENTOS

A equipe SIRSoccer agradece a Faculdade de Educação Tecnológica do Estado do Rio de Janeiro (FAETERJ - Petrópolis) pelo apoio.

REFERENCES

[1] Edson Bernardes Ferreira Filho. *Desenvolvimento de um Time de Futebol de Robôs Categoria IEEE Very Small Size*. PhD thesis, Universidade Federal de Ouro Preto, 2013.

[2] Stefan Zickler, Tim Laue, Oliver Birbach, Mahisorn Wongphati, and Manuela Veloso. Ssl-vision: The shared vision system for the robocup small size league. In *RoboCup 2009: Robot Soccer World Cup XIII*, pages 425–436. Springer, 2009. url.

[3] Fred Turek. Machine vision fundamentals, how to make robots see. *NASA Tech Briefs magazine*, 35(6):60–62, 2011.

[4] Gary Bradski et al. The opencv library. *Doctor Dobbs Journal*, 25(11):120–126, 2000.

[5] Bjarne Stroustrup. *Why C++ is not just an object-oriented programming language*, volume 6. ACM, 1995.

[6] Johnathan Fercher da Rosa. Construção de um time de futebol de robôs para a categoria ieee very small size soccer, 2015.

[7] Hiroki Ohta, Shinji Ozawa, Minami Miyauchi, Hajime Takata, Akira Sonoda, and Hisami Iri. Computer classification of rosette-forming cells from microscope images. *Systems and computers in Japan*, 18(4):64–75, 1987.

[8] GLUT - The OpenGL Utility Toolkit. http://www.opengl.org/resources/libraries/glut/. Accessado: 2014.07.30.

[9] Real-time physics simulation. http://bulletphysics.org/wordpress/. Acessado: 2016.05.23.

[10] Bruno Siciliano, Oussama Khatib, and Frans Groen. *Springer Tracts in Advanced Robotics*. Springer, 2002.

[11] Saso Blazic. On periodic control laws for mobile robots. *Industrial Electronics, IEEE Transactions on*, 61(7):3660–3670, 2014.

[12] ZigBee Alliance. https://www.zigbee.org. Accessado: 2012.04.09.

[13] Xbee/xbee-pro rf modules. http://www.digi.com/lp/xbee. Accessado: 2013-05-07.

[14] Arduino: Arduino homepage. http://www.arduino.cc. Accessado: 2013-05-07.

[15] Toshiba bi-cd integrated circuit. http://www.datasheetspdf.com/datasheet/TB6614FNG.html. Accessado: 2013-05-07.

[16] Pololu: Robotics & Eletronics. http://www.pololu.com/catalog/product/2361. Accessado: 2013.05.07.

[17] Pololu HomePage. https://www.pololu.com/product/2591. Accessado: 2015.04.13.

[18] Pololu HomePage. https://www.pololu.com/product/713. Accessado: 2015.04.13.

[19] ARDUINO: Arduino Learning. http://arduino.cc/en/Tutorial/PWM. Accessado: 2014.05.29.

[20] DFRobots HomePage. http://www.dfrobot.com/index.php. Accessado: 2013.05.07.